(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
25 January 2007 (25.01.2007)

PCT

(10) International Publication Number
WO 2007/009254 A1

(51) International Patent Classification:
G06F 7/00 (2006.01) *G06F 12/02* (2006.01)
G06F 7/02 (2006.01) *G06F 17/30* (2006.01)

(21) International Application Number:
PCT/CA2006/001202

(22) International Filing Date: 21 July 2006 (21.07.2006)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
2,513,010 22 July 2005 (22.07.2005) CA

(71) Applicant *(for all designated States except US)*: **RESEARCH IN MOTION LIMITED** [CA/CA]; 295 Phillip Street, Waterloo, Ontario N2L 3W8 (CA).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: **KNOWLES, Michael** [CA/CA]; 847 Brandenburg Blvd., Waterloo, Ontario N2T 2X5 (CA). **TAPUSKA, David** [CA/CA]; 712 Willow Wood P1, Waterloo, Ontario N2T 2T7 (CA). **KALOUGINA, Tatiana** [CA/CA]; 124 Keats Way P1, Waterloo, CA N2L 5H3 (CA).

(74) Agents: **PERRY, Stephen, J.** et al.; PERRY + PARTNERS, 1300 Yonge Street, Suite 500, Toronto, Ontario M4T 1X3 (CA).

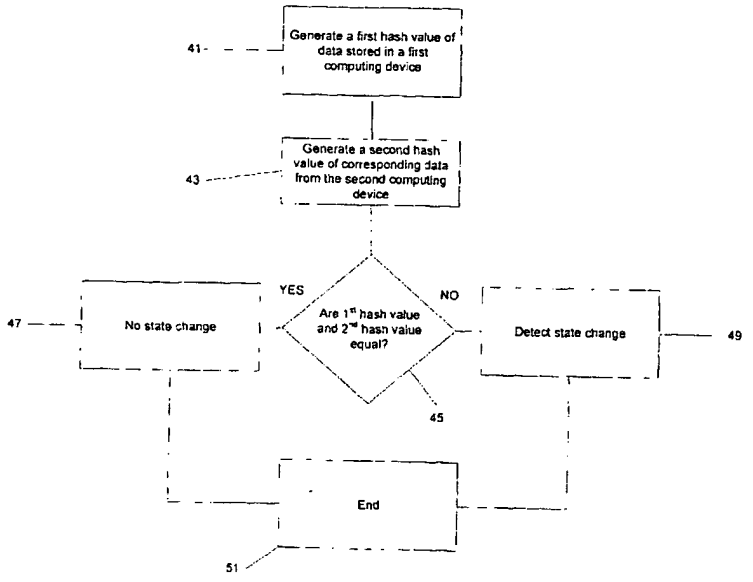
(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LV, LY, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: A METHOD FOR DETECTING STATE CHANGES BETWEEN DATA STORED IN A FIRST COMPUTING DEVICE AND DATA RECEIVED FROM A SECOND COMPUTING DEVICE



(57) Abstract: A method for detecting state changes between data stored in a first computing device and data retrieved from a second computing device includes: generating a first hash value of the data stored in the first computing device; generating a second hash value of corresponding data retrieved from the second computing device; comparing the first hash value to the second hash; and detecting a state change in the event of a difference therebetween.

WO 2007/009254 A1

A Method for Detecting State Changes Between Data Stored in a First Computing Device and Data Received from a Second Computing Device

Field

[0001] This specification relates generally to mobile data communication systems, and more particularly to a method for detecting state changes between data stored in a first computing device and data retrieved from a second computing device.

Background

[0002] Mobile communication devices are becoming increasingly popular for business and personal use due to a relatively recent increase in number of services and features that the devices and mobile infrastructures support. Handheld mobile communication devices, sometimes referred to as mobile stations, are essentially portable computers having wireless capability, and come in various forms. These include Personal Digital Assistants (PDAs), cellular phones and smart phones.

[0003] It is known in the art to provide Internet browser functionality in such mobile communication devices. In operation, a browser user-agent in the handheld mobile communication device issues commands to an enterprise or proxy server implementing a Mobile Data Service (MDS), which functions as an acceleration server for browsing the Internet and transmitting text and images to the mobile device for display. Such enterprise or proxy servers generally do not store the state of their clients (i.e. the browser user-agent), or if they do, the state that is stored is minimal and limited to HTTP state (i.e. cookies). Typically, such enterprise or proxy servers fetch and transmit data to the browser user-agent when the browser makes a data request. In order to improve the performance of the browser on the mobile device, some enterprise or proxy servers fetch all the data required in order to fulfill the data request from the browser, aggregate the fetched data, and transmit the data to the device browser. For instance, if a HyperText Markup Language (HTML) page is requested, the enterprise or proxy server fetches any additional files referenced within the HTML page (e.g. Images, inline CSS code, JavaScript, etc.). Since the proxy server fetches all the additional files within the HTML file, the device does not have to make additional data requests to retrieve these additional files. Although this methodology is faster than having the device make multiple requests, the proxy server nonetheless has to send all of the data again if the site is later revisited. This is because the proxy server has no knowledge of the device caches (e.g. caches that are saved in persistent memory, for different types of data such as a content cache to store raw data that is cached as a result of normal browser activity, a channel cache containing data

that is sent to the device by a channel or cache push, and a cookie cache containing cookies that are assigned to the browser by visited Web pages). For example, if a user browses to CNN.com, closes the browser to perform some other function (e.g. place a telephone call or access e-mail messages, etc.) and then later accesses the CNN.com Web site (or follows a link from CNN.com to a news story), the banner "CNN.com" will be transmitted from the MDS to the device browser each time the site is accessed, thereby consuming significant bandwidth, introducing delay, etc.

[0004] It is known in the art to provide local file caching. One approach is set forth in *GloMop: Global Mobile Computing By Proxy*, published September 13, 1995, by the GloMop Group, wherein PC Card hard drives are used as portable file caches for storing, as an example, all of the users' email and Web caches. The user synchronizes the file caches and the proxy server keeps track of the contents. Mobile applications (clients) are able to check the file caches before asking for information from the proxy server by having the server verify that the local version of a given file is current.

Summary

[0005] In general, there is provided a method for detecting state changes between data stored in a first computing device and data retrieved from a second computing device. The method includes: generating a first hash value of the data stored in the first computing device; generating a second hash value of corresponding data retrieved from the second computing device; and comparing the first hash value to the second hash value and detecting a state change in the event of a difference there between.

[0006] A specific application of this method provides for communicating information between the second computing device, such as an enterprise or proxy server and the first computing device, such as a mobile Internet browser. An HTTP-like protocol is set forth, referred to herein as the Browser Session Management (BSM) protocol, for providing a control channel between the second computing device, and the first computing device, so that the first computing device can communicate to the second computing device what data the first computing device has stored in memory (from previous browsing). The BSM protocol is an "out of band" protocol in that BSM communications are in addition to the usual stream of HTTP requests from the first computing device to the second computing device, and provide "metadata" relating to cache contents. This metadata is used by the second computing device when handling subsequent requests from the first computing device, to determine what data to send to the first computing device, thereby significantly reducing data transfer on subsequent requests relative to the prior

art methodology discussed above.

[0007] Because the second computing device is aware of what the first computing device has stored in its cache, the amount of data sent to the first computing device may be reduced, thereby increasing the performance of the first computing device and reducing operational cost. For the application given above wherein the first computing device is a mobile device browser and the second computing device is a proxy server, if after the first request the CNN.com banner is cached and if the proxy server "knows" that the information has been cached then there will be no need to send the CNN.com banner to the mobile device browser upon subsequent visits to the CNN Web site.

[0008] According to another aspect, messages from the device to the proxy server contain hash values of different portions of documents (rather than the actual URLs) which are used by the proxy server to detect state changes in the device and utilize the information in preparing documents for transmission to the device. In another embodiment, the device sends hashes of the actual data of the portions (i.e. the actual image data, JavaScripts, StyleSheets, etc.) and the proxy server compares the received and stored data hashes for the portions to determine if the device already has the data for a particular portion (e.g. previously retrieved with a different URL), in which case the proxy server sends a response to the device with a header that indicates the device already has the data that is to be used for that portion. A person of skill in the art will appreciate that a one-way hash function transforms data into a value of fixed length (hash value) that represents the original data. Ideally, the hash function is constructed so that two sets of data will rarely generate the same hash value. Examples of known hash functions include MD2, MD5 and SHA-1.

[0009] In contrast to the prior art *GloMop* caching methodology discussed above, the exemplary method set forth herein synchronizes the cache contents when the mobile device browser connects to the proxy server in order to initiate a session and keeps track of changes to the cache via knowledge of what data has been sent to the mobile device browser in combination with state information periodically received from the mobile device browser identifying what has actually been cached. Also, as set forth in greater detail below, the proxy server uses this cache knowledge to determine what to send back to the mobile device browser. In contrast, the prior art *GloMop* methodology does not contemplate sending any state information to the proxy server for identifying what has actually been cached in the device. Moreover, the prior art *GloMop* approach first checks the local cache, and then queries the proxy server to determine whether a particular data item in the cache is current or not.

According to the *GloMop* prior art, the proxy server does not use its own knowledge of the mobile device browser cache to determine what to send back to the mobile device browser.

[0010] Additional aspects and advantages will be apparent to a person of ordinary skill in the art, residing in the details of construction and operation as more fully hereinafter described and claimed, reference being had to the accompanying drawings.

Brief Description of the Drawings

[0011] A detailed description of the preferred embodiment is set forth in detail below, with reference to the following drawings, in which:

[0012] Figure 1 is a block diagram of a communication system for implementing Internet browsing functionality in a mobile communication device;

[0013] Figure 2A shows communication protocol stacks for the communication system of Figure 1;

[0014] Figure 2B shows communication protocol stacks for a Browser Session Management (BSM) protocol according to an exemplary embodiment;

[0015] Figure 3 is a flowchart showing the method for communicating information between a proxy server and a mobile Internet browser, according to the preferred embodiment; and

[0016] Figure 4 is a flowchart of an exemplary method according to the present specification.

Detailed Description

[0017] Figure 1 depicts the architecture of a system for providing wireless e-mail and data communication between a mobile device 1 and an enterprise or proxy server 9. Communication with the device 1 is effected over a wireless network 3, which in turn is connected to the Internet 5 and proxy server 9 through corporate firewall 7 and relay 8. Alternatively, the device 1 can connect directly (via the Internet) through the corporate firewall 7 to the proxy server 9. When a new message is received in a user's mailbox within email server 11, enterprise or proxy server 9 is notified of the new message and email application 10 (e.g. Messaging Application Programming Interface (MAPI), MS Exchange, etc.) copies the message out to the device 1 using a push-based operation. Alternatively, an exemplary architecture for proxy server 9 may provide a browsing proxy but no email application 10. Indeed, the exemplary embodiment set forth herein relates to mobile browser device functionality and is not related to email functionality. Proxy server 9 also provides access to data on an application server 13 and the

Web server 15 via a Mobile Data Service (MDS) 12. Additional details regarding e-mail messaging, MAPI sessions, attachment service, etc., are omitted from this description as they are not germane. Nonetheless, such details would be known to persons of ordinary skill in the art.

[0018] In terms of Web browsing functionality, the device 1 communicates with enterprise or proxy server 9 using HTTP over an IP protocol optimized for mobile environments. In some embodiments, the device 1 communicates with the proxy server 9 using HTTP over TCP/IP, over a variant of TCP/IP optimized for mobile use (e.g. Wireless Profiled TCP), or over other, proprietary protocols. For example, according to the communications protocol of Fgure 2A, HTTP is run over Internet Point-to-Point Protocol (IPPP) and an encrypted Global Messaging Exchange (GME) channel over which datagrams are exchanged to transport data between the device 1 and proxy server 9. The GME datagrams are 64Kbit in size whereas the wireless network 3 can only transport UDP datagrams with payloads up to 1500 bytes. Therefore, a Message Delivery Protocol (MDP) is used to separate the GME datagrams into one or more MDP packets, each of which is less than 1500 bytes (default size 1300 bytes), which are transported over UDP/IP to and from the relay 8 which, in turn communicates with the proxy server 9 via Server Relay Protocol (SRP)/TCP/IP. The MDP protocol includes acknowledgements, timeouts and re-sends to ensure that all packets of the GME datagram are received.

[0019] The communication between the device 1 and proxy server 9 is optionally encrypted with an encryption scheme, such as Triple Data Encryption Algorithm (TDEA, formerly referred to as Triple Data Encryption Standard (Triple DES)), as is known in the art. The proxy server 9 enables Internet access, preprocesses and compresses HTML and XML content from the Web server 15 before sending it to the device 1, transcodes content type, stores HTTP cookies on behalf of the device 1, and supports certificate authority authentications, etc.

[0020] In response to a request from the device browser, the proxy server 9 retrieves content from Web server 15 and creates a custom document containing both images to be displayed on the device and data in the form of compressed versions of requested portions of the document. The document is preferably of "multi-part" format to improve transmission to and processing efficiency within the device 1. Specifically, in order to display composite Web pages (i.e. pages composed of a main WML or HTML page and one or more related auxiliary files, such as style sheets, JavaScript files, or image files) the device browser is normally required to send multiple HTTP requests to the proxy server 9. However, according to the multi-part

generation feature, the proxy server 9 posts all necessary parts of a composite Web page in a single bundle, enabling the browser to download all the required content with a single request. The header in the server response identifies the content as a multi-part bundle (e.g. Multi-Purpose Mail Extensions (MIME)/multipart,as defined by RFC 2112, E. Levinson, March 1997).

[0021] In order to indicate device browser state information to the proxy server 9, three transitional state messages are defined herein, as follows: CONNECT, UPDATE and DISCONNECT, each of which conforms to the exemplary BSM protocol. As shown in Figure 2B, the BSM communications protocol is identical to the protocol of Figure 2A except that the conventional HTTP layer of the protocol stack is replaced by an HTTP-like BSM layer.

[0022] The CONNECT transitional message creates a new session with a connection identifier carried in the payload, device information and state data (e.g. current cache and device information) in the form of a set of hash functions for use by the proxy server 9 in preparing a response. Specific care is taken not to identify to the proxy server 9 what cookies or cache entries are contained on the device 1. Only hash values of the state data are sent to the proxy server 9 in order to protect the identity of state data on the device 1.

[0023] The CONNECT message also contains a unique authentication key for generating a MAC (Message Authentication Code) using a Hash Message Authentication Code (HMAC) algorithm that incorporates a cryptographic hash function in combination with the authentication key. Each portion of a multi-part document from the proxy server 9 also contains an HMAC, generated using the authentication key, that is used for authenticating the proxy server 9 before adding that portion to the device cache. This prevents a third party from creating its own multi-part document and sending it to the device 1 for injecting cache entries that could be used to extract personal information from the user.

[0024] Upon receipt of the CONNECT message, the proxy server 9 uses the state information to regulate or control the transmission of content retrieved from Web server 15 (step 23) to the device 1. One example of an application where this information can be used is when the proxy server 9 is pre-fetching images, inline Cascading Style Sheets (CSS), JavaScript, and the like for an HTML document. If the proxy server 9 already knows that the device 1 has the image, inline CSS, or JavaScript document, there is no need for resending the documents.

[0025] The UPDATE transition message notifies the proxy server 9 of changes that have occurred on the device 1 since the last CONNECT message or the last UPDATE message, between the device 1 and proxy server 9 (e.g. new cache entries added because of a push, or

invoking the "Low Memory Manager" (LMM) or other memory-space preservation policies on the device and purging items from the cache).

[0026] The DISCONNECT transition message notifies the proxy server 9 that the device 1 will no longer send any more messages using the connection identifier specified in the payload. The proxy server 9 can then de-allocate any memory reserved for the connect session between the device 1 and proxy server 9. Upon receiving the disconnect message, the proxy server 9 deletes any session cookies for the device 1 (if it is processing cookies) along with state information. Receiving a request on the identified connection after the DISCONNECT has been received, and before any subsequent CONNECT message has been received, is defined as an error.

[0027] Since state is indicated from the device 1 to the proxy server 9, and state may be stored in transient memory within proxy server 9, a mechanism is provided for the proxy server 9 to return to the device 1 a message indicating that the session the device is trying to use is not valid. Once this occurs, the device 1 issues a new CONNECT message and establishes a new session with the proxy server 9, and re-issues the original request.

[0028] The data protocol set forth herein is similar to HTTP in order to reduce complexity and to reuse code that already exists for the HTTP protocol. Thus, data transmission according to this protocol begins with a STATE keyword; followed by a BSM (Browser Session Management) protocol identifier and a "Content-Length" header. The end of the "headers" is indicated by a double CRLF (a sequence of control characters consisting of a carriage return (CR) and a line feed (LF)), much like HTTP. After the double CRLF pair (i.e. \r\n) a WBXML (WAP Binary Extensible Markup Language) encoded document is inserted as the message payload. The WBXML document is later decoded using a DTD (Document Type Definition) and codebook, as discussed in greater detail below. The indication of the protocol version refers to what version of the DTD to validate the request against (ie. BSM/1.1 stipulates using version 1.1 of the DTD). It should be noted that WBXML encoding of the contents of BSM messages is set forth to allow for more efficient processing of the BSM message at the device 1, but that in alternate embodiments, the BSM message may be formatted as normal (textual) XML.

[0029] The following is an example communication using the protocol according to the preferred embodiment:

```
CONNECT BSM/1.0\r\n
Content-Length: 40\r\n
\r\n
```

<WBXML Encoded document of length 40 bytes>

BSM/1.0 200\r\n
r\n

[0030] In the foregoing, the first four lines form the CONNECT message from the device 1 to the proxy server 9, and the last two lines are the response from the proxy server 9.

[0031] An exemplary XML document, is as follows:

```
<?xml version="1.0"?>
<!DOCTYPE bsm PUBLIC "-// DTD BSM 1.0//EN"
          "http://www.something.com/go/mobile/BSM/bsm_1.0.xml">
<bsm id="2" hmac="12345678901234567890">
<cache>
<size>123012</size>
<entry urlHash="FEEDDEED01" dataHash="FDDEDEED11" etag="SomeEtag"
expiry="256712323"/>
</cache>
<device>
<version>4.0.1.123</version>
<memfree>12342342</memfree>
</device>
</bsm>
```

[0032] In the example, the state data includes the URL of an HTML page within the device cache. It will be noted that the XML document payload includes a connection identifier (i.e. bsm id="2"), a value indicating when the document was last modified (i.e. etag="SomeEtag"), a page expiry (i.e. expiry="256712323"), and hash values for a URL (i.e. entry urlHash="FEEDDEED01") and a data attribute (i.e. entry dataHash="FDDEDEED11") rather than transmitting the actual URL and data attribute themselves. Thus, as shown in Figure 3, the hashes of the URL and data attribute of the cached page are sent to the proxy server 9 in the CONNECT string (step 21). The proxy server 9 then fetches the requested page from Web server 13 (step 23), computes hashes of device browser state data (step 25) and data from the Web server 13 (step 27), and compares the hashes of the URL and data attribute of the requested page with the hashed URL and data attribute of the cached page, and also compares the time stamps/expiration information (step 29) in order to determine whether the cached page is current. Specifically, in response to the proxy server 9 retrieving a portion from the Web server 13, it computes the dataHash and urlHash of that portion and performs a comparison to the dataHashes and urlHashes of the entries it has saved. There are three cases.

[0033] In the first case, if both the dataHash and the urlHash of the retrieved portion match the dataHash and urlHash of a cache entry that the proxy server 9 knows the device 1 has, then the server 13 simply omits this portion from the response, as the device 1 still has a valid entry

in its cache.

[0034] In the second case, if the dataHash of the retrieved portion matches the dataHash of a cache entry that the proxy server 9 knows the device 1 has, but the urlHash of the retrieved portion does not match the urlHash of that cache entry, the server 13 inlines this updated portion in the combined response to the device 1. However, because the dataHash matches a dataHash of an entry that already exists on the device 1, the inlined response does not include the actual data, but instead only includes a new HTTP header whose value is the new dataHash. When the device 1 receives this inlined portion, it detects the special header, looks for the cache entry with that dataHash, and either creates or updates its cache entry for that URL with the data corresponding to the dataHash by copying that data from the other cache entry (the cache for device 1 is modified to have two indexes, one to retrieve cache entries by URL, the other to retrieve cache entries by dataHash). Finally, if the proxy server 9 already has a cache entry for the urlHash, it updates that entry with the new dataHash; otherwise it creates a new entry for this portion.

[0035] In the third case, if the dataHash of the retrieved portion does not match the dataHash of any of the cache entries that the proxy server 9 has received from the device 1 in the BSM messages, then the server inlines the entire portion (headers and new data), since this portion has been updated and the device 1 does not contain the updated value anywhere in its cache.

[0036] Although not indicated in Figure 3, it will be appreciated that each inline part to be added to a document to be displayed at the device 1 is fetched. If the response code from the proxy server indicates a "304" (step 31), then the part (i.e., the "304" response) is written as a block in the multipart document. On the other hand, if the proxy server 9 returns a "200" (step 33), then the hash compare operation is performed, and the portion is only included in the multipart document if the hash compare function indicates it is not already on the device 1.

[0037] An exemplary DTD, according to the preferred embodiment, is as follows:

```
<!ELEMENT bsm (cache?, device)>
<!ATTLIST bsm
          id       NMTOKEN       #REQUIRED
>
<!ELEMENT cache (size, (entry)+)>
<!ATTLIST cache
          action   (add|remove|remove_all|quick_add)   "add"
>

<!ELEMENT entry EMPTY>
<!ATTLIST entry
```

```
          urlHash        CDATA       #REQUIRED
          dataHash       CDATA       #REQUIRED
          etag           CDATA       #IMPLIED
          expiry         NMTOKEN     #IMPLIED
          size           NMTOKEN #IMPLIED
          last-modified NMTOKEN #IMPLIED
>
<!ELEMENT size (#PCDATA)>
<!ELEMENT device (version, memfree)>
<!ELEMENT version (#PCDATA)>
<!ELEMENT memfree (#PCDATA)>
<!ELEMENT hmac (#PCDATA)>

Element/Code
HMAC 12

Attribute/Code
size 9 (instead of action)
lastModified 10
actionAdd 11
actionRemove 12
actionRemoveAll 13
actionQuickAdd 14
```

[0038] Finally, an exemplary codebook, is as follows:

Element	Code
Session	5
Cache	6
Size	7
Entry	8
Device	9
Version	10
MemFree	11
HMAC	12

Attribute	Code
Id	5
UrlHash	6
dataHash	7
ETag	8
Expiry	9
Action	10

[0039] As is well known in the art, the codebook is used as a transformation for compressing the XML document to WBXML, wherein each text token is represented by a single

byte from the codebook.

[0040] As discussed above, the proxy server 9 transmits multi-part documents in a proprietary format of compressed HTML, interspersed with data for images and other auxiliary files (which may or may not be related to the main HTML Web page). However, in a departure from conventional HTML, each document part may also include a response code (e.g. "200" for OK, or "304" for "not modified" to indicate that the specified document part has already been cached in the device 1). This may be used for selective downloading of document parts rather than entire documents and for indicating when a part (e.g. image) is about to expire. This is useful, for example, when one Web page links to another page containing one or more common elements.

[0041] Of course, certain device requests (e.g. page refresh) will always result in a full document download, irrespective of device state information stored in the proxy server 9.

[0042] It is contemplated that the inclusion of response codes may be used by heuristic processes within the proxy server 9 to learn user behaviour and modify downloading of documents based on tracking the history of certain changes reflected in the hash value (e.g. the server 9 may learn to download a certain page (e.g. CNN news) at a particular time each day based the user's history of issuing requests for that page at regular times. As discussed above, because the downloaded documents are multi-part and contain embedded response codes, only those portions of the document that have changed are actually downloaded.

[0043] Figure 4 illustrates a broad aspect of the exemplary method, wherein a first hash value is generated in a first computing device, such as mobile device 1 (step 41), and a second hash value is generated in a second computing device, such as proxy server 9 (step 43). The first and second hash values are then compared (step 45). If the hash values are identical (step 47), no change of state is detected between the data stored in the first and second computing devices. On the other hand, if the hash values are identical (step 49), state change is detected between the data stored in the first and second computing devices. The method then ends (step 51).

[0044] As indicated above, the protocol of the preferred embodiment is preferably carried over a proprietary IPPP transport layer, but can also be easily adapted to run over TCP/IP on a specific port. The protocol is preferably implemented as a handler in the proxy server 9, thereby simplifying any currently existing protocol. (e.g. to avoid overloading a current HTTP protocol).

[0045] A person skilled in the art, having read this description of the preferred embodiment,

may conceive of variations and alternative embodiments. For example, the conditional transfer of data based on communication of state information, as set forth above, may also be applied to separately transmitting individual portions of the multipart document as opposed to transmitting the entire document at once.

[0046] In some embodiments, the proxy server 9 uses heuristic algorithms to learn what additional data requests the device may make based on knowledge of the current request, and knowledge of past activity. In some instances, the device may follow a pattern of requesting a first Web page, and then a second Web page. For example, the device may first request the "cnn.com" Web page, and then request the "cnn.com/news" Web page. The proxy server 9 learns this pattern, and whenever the device requests the first Web page, the proxy server 9 determines that the device is likely to then request the second Web page. The proxy server 9 then fetches the second Web page, and uses its knowledge of the data cached on the device 1 (i.e. from the state information transferred to the proxy server 9 during initiation of the present connection) to determine whether the second Web page already exists within the data cached on the device. If so, the proxy server 9 includes information about the second Web page via response codes embedded within the response provided for the first Web page. If the device 1 requires the second Web page, then the device 1 can reference its cache and can avoid having to make a request to the proxy server 9 for the second Web page.

[0047] In other embodiments, heuristic processes within the proxy server 9 learn user behaviour and modify downloading of documents based on tracking the history of certain changes reflected in the hash value (e.g. the proxy server 9 may learn to download a certain page (e.g. CNN news) at a particular time each day based the user's history of issuing requests for that page at regular times). As discussed, because the downloaded documents are multi-part and contain embedded response codes, only those portions of the document that have changed are actually downloaded.

[0048] All such variations and alternative embodiments are believed to be within the ambit of the claims appended hereto.

What is claimed is:

1. A method for detecting state changes between data stored in a first computing device and data retrieved from a second computing device, comprising:

generating a first hash value of said data stored in said first computing device;

generating a second hash value of corresponding data retrieved from said second computing device; and

comparing said first hash value to said second hash value at said second computing device and detecting a state change in the event of a difference therebetween.

2. The method of claim 1, wherein said first hash value represents state information of said first computing device and said second hash value represents said corresponding data retrieved from said second computing device.

3. The method of claim 2, further comprising transmitting said first hash value from said first computing device to said second computing device, and transmitting said corresponding data from said second computing device to said first computing device only in the event said first hash value and said second hash value are different.

4. The method of claim 3, further comprising embedding said first hash value within a transitional state message from said first computing device to said second computing device.

5. The method of claim 4, wherein said first hash value is generated by performing a hash function on a URL being requested by said first computing device, and said second hash value is generated by performing a hash function on a corresponding URL stored at said second computing device.

6. The method of claim 4, wherein said first hash value is generated by performing a hash function on a data attribute being requested by said first computing device, and said second hash value is generated by performing a hash function on a corresponding data attribute stored at said second computing device.

7. A method for detecting state changes between data stored in a first computing device and data retrieved from a second computing device, comprising:

generating a first pair of hash values of said data stored in said first computing device;

generating a second pair of hash values of corresponding data retrieved from said second computing device; and

comparing said first pair of hash values to said second pair of hash values at said second computing device and detecting a state change in the event of a difference therebetween.

8. The method of claim 7, wherein said first pair of hash values represents state information regarding a document component stored in said first computing device and said pair of second hash values represents corresponding data retrieved from said second computing device.

9. The method of claim 8, further comprising transmitting said first pair of hash values from said first computing device to said second computing device, and transmitting said corresponding data from said second computing device to said first computing device only in the event said first pair of hash values differ from said second pair of hash values.

10. The method of claim 9, further comprising embedding said first pair of hash values within a transitional state message from said first computing device to said second computing device.

11. The method of claim 9, wherein said pair of first hash values is generated by respectively applying a hash function on a URL and on a data attribute being requested by said first computing device, and said second pair of hash values is generated by respectively applying a hash function on a corresponding URL and on a corresponding data attribute stored at said second computing device.

12. The method of claim 11, further comprising detecting that the hash function of said corresponding data attribute stored at said second computing device matches the hash function of a cache entry of said data attribute in said first computing device, detecting that the hash function of said corresponding URL stored at said second computing device does not match the hash function of a cache entry of said URL in said first computing device, and sending an inlined response from said second computing device to said first computing device that does not include said corresponding data attribute but does include a HTTP header whose value is the hash function of said corresponding data attribute stored at said second computing device.

13. The method of claim 12, further comprising receiving said inlined response within said first computing device, detecting said header and in response locating a first cache entry with said hash function of said corresponding data attribute within said first computing device, and one of either creating or updating a further cache entry for said URL with data corresponding to said hash function of said corresponding data attribute by copying from said first cache entry.

14. The method of claim 12, further comprising locating a first cache entry with said hash function of said corresponding data attribute within said second computing device, and one of either creating or updating a further cache entry for said URL with data corresponding to said hash function of said corresponding data attribute by copying from said first cache entry.

15. The method of claim 11, further comprising detecting that the hash function of said corresponding data attribute stored at said second computing device does not match the hash function of a cache entry of said data attribute in said first computing device, and sending an inlined response from said second computing device to said first computing device for updating said corresponding data retrieved from said second computing device, wherein said inlined response includes the corresponding data attribute retrieved from said second computing device.

16. The method of any one of the foregoing claims, wherein said first and second hash values are generated using hash functions selected from a group comprising MD2, MD5 and SHA-1.

16. A system for detecting state changes between data stored in a first computing device and data retrieved from a second computing device, comprising:

a first hash value generator for generating a first hash value of said data stored in said first computing device;

a second hash value generator for generating a second hash value of corresponding data retrieved from said second computing device; and

a comparator for comparing said first hash value to said second hash value and detecting a state change in the event of a difference therebetween.

17. A system for detecting state changes between data stored in a first computing device and data retrieved from a second computing device, comprising:

a first hash value generator for generating a first pair of hash values of said data stored in said first computing device;

a second hash value generator for generating a second pair of hash values of corresponding data retrieved from said second computing device; and

a comparator for comparing said first pair of hash values to said second pair of hash values and detecting a state change in the event of a difference therebetween.

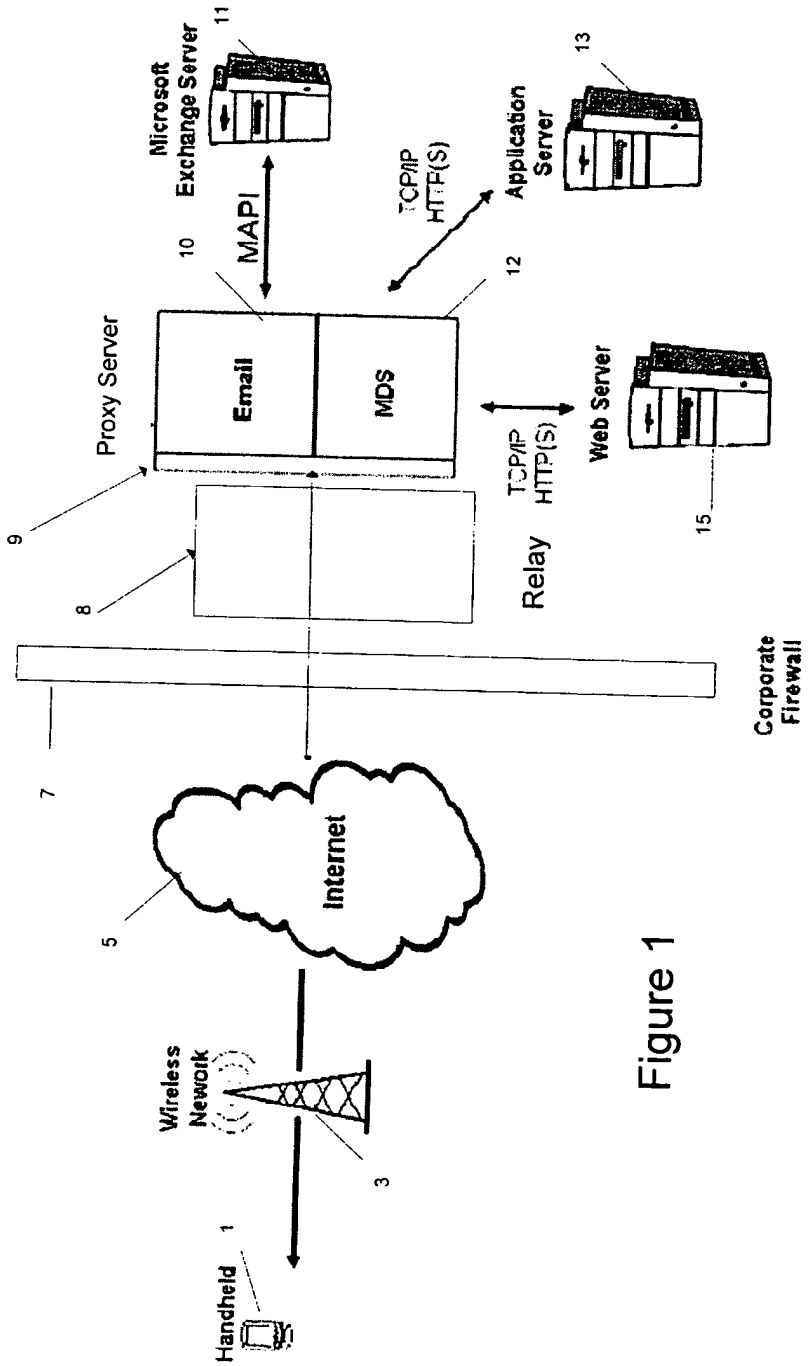

Handheld
1
Wireless Network
3
Internet
5
7
Corporate Firewall
Relay
8
9
Proxy Server
Email
MDS
10
MAPI
Microsoft Exchange Server
11
TCP/IP HTTP(S)
12
Application Server
13
TCP/IP HTTP(S)
Web Server
15

Figure 1

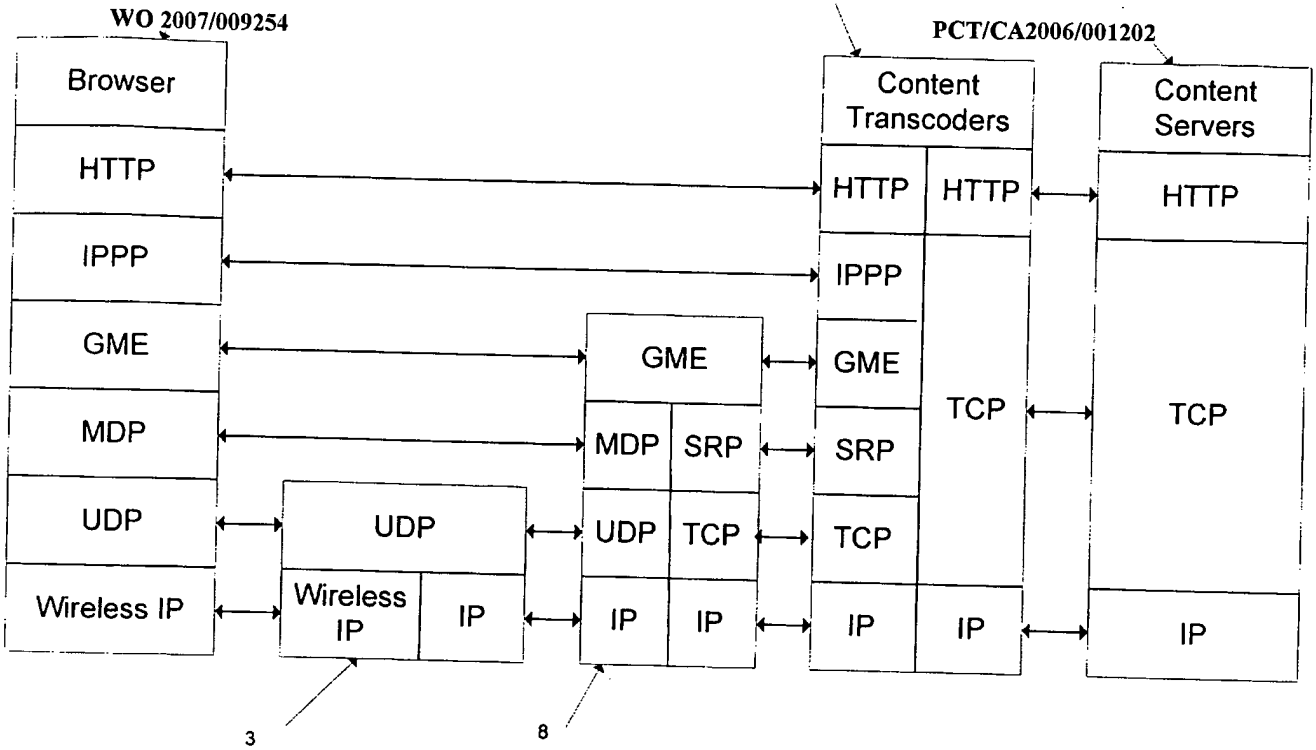


Figure 2A

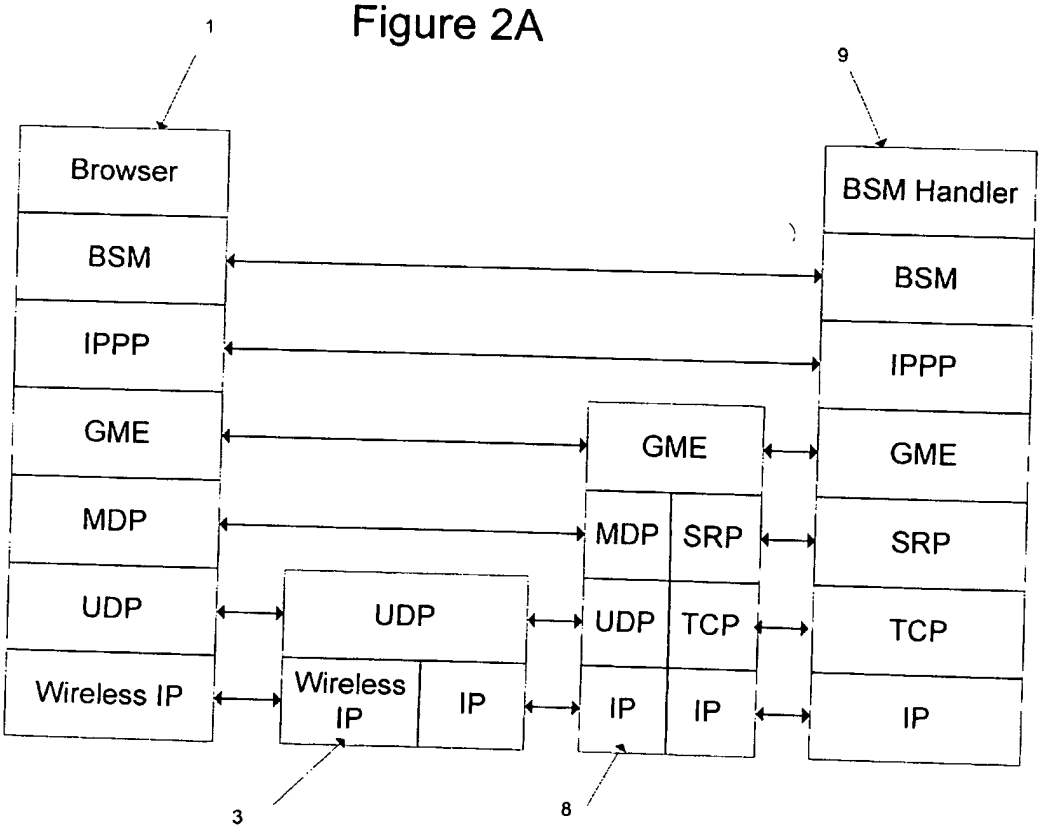


Figure 2B

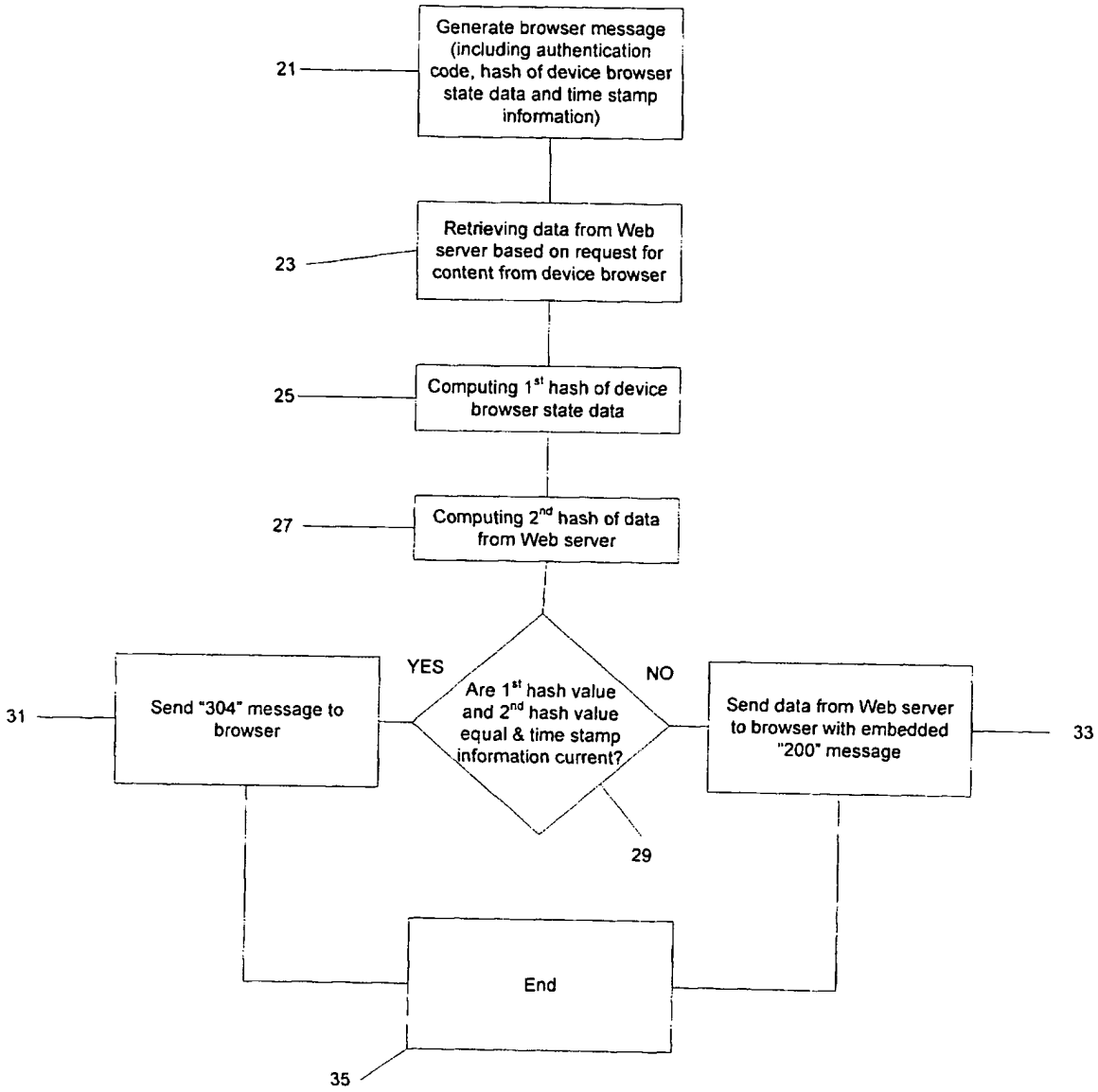

Generate browser message (including authentication code, hash of device browser state data and time stamp information)
21
Retrieving data from Web server based on request for content from device browser
23
Computing 1st hash of device browser state data
25
Computing 2nd hash of data from Web server
27
Are 1st hash value and 2nd hash value equal & time stamp information current?
YES
NO
29
Send "304" message to browser
31
Send data from Web server to browser with embedded "200" message
33
End
35

Figure 3

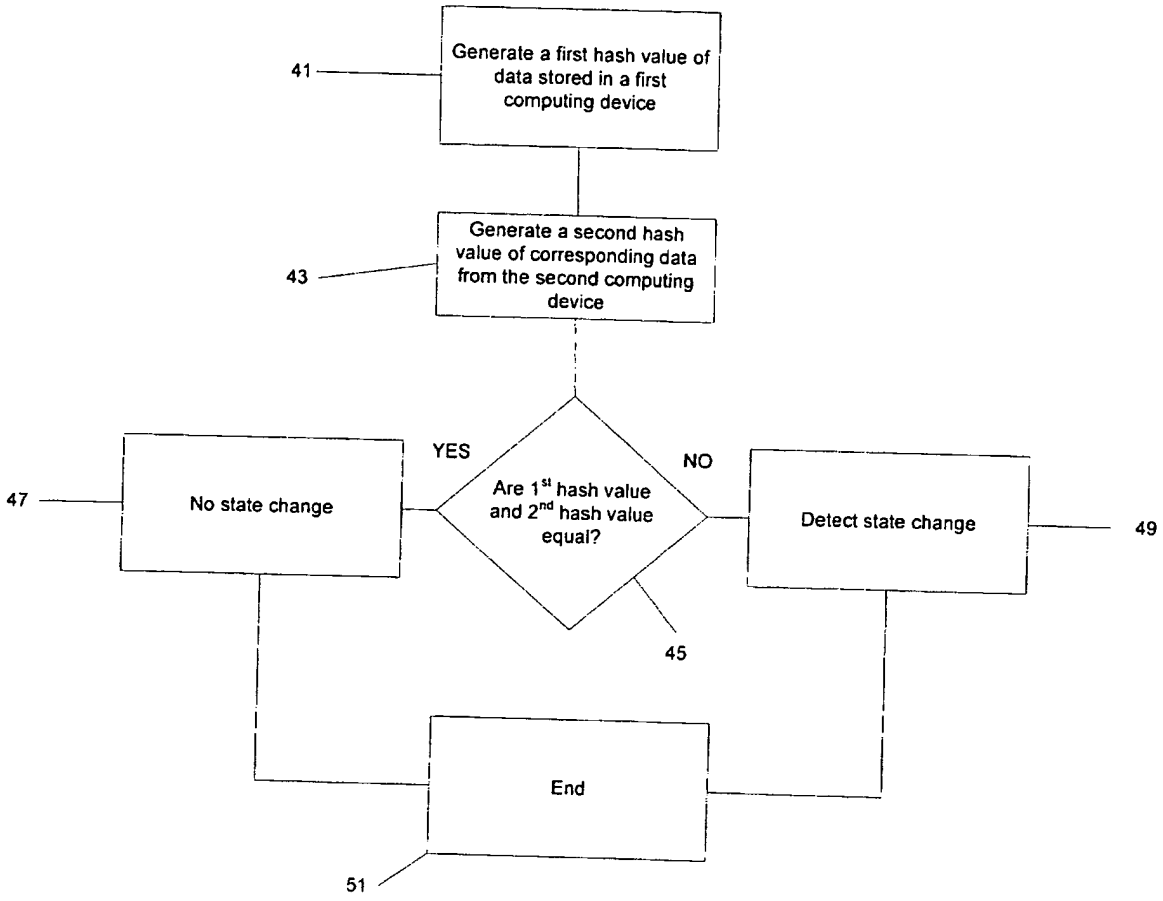

Generate a first hash value of data stored in a first computing device
41
Generate a second hash value of corresponding data from the second computing device
43
Are 1st hash value and 2nd hash value equal?
YES
NO
45
No state change
47
Detect state change
49
End
51

Figure 4

INTERNATIONAL SEARCH REPORT

International application No.
PCT/CA2006/001202

A. CLASSIFICATION OF SUBJECT MATTER
IPC: ***G06F 7/00*** (2006.01) , ***G06F 7/02*** (2006.01) , ***G06F 12/02*** (2006.01) , ***G06F 17/30*** (2006.01)
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC: *G06F 7/00* (2006.01); *G06F 7/02* (2006.01)

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic database(s) consulted during the international search (name of database(s) and, where practicable, search terms used)
Databases searched: Canadian Patent Database; USPTO WEST (full-text patent database, pre-grant publication, EPO/JPO abstracts); Esp@cenet; and, Internet.
Search words used: first hash value, second hash value, first and second computing device; mobile data service, Internet, communications, MD2, MD5, SHA-1, wireless, URL, browser, cache, and hash codes.

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 6,286,032 (Oberlander et al.) 4 September 2001 (04.09.01) entire document	1 to 11 and 15 to 18
Y	US 2001/0027450 (Shinoda et al.) 4 October 2001 (04.10.01) entire document	7, 15, and 18
Y	US 2003/0120647 (Aiken et al.) 26 June 2003 (26.06.03) page 1, par. 80	5, 11, and 15
Y	US 2004/0220975 (Carpentier et al.) 4 November 2004 (04.11.04) page 3, par. 35	16

[X] Further documents are listed in the continuation of Box C. [X] See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
25 October 2006 (25-10-2006)	1 November 2006 (01-11-2006)
Name and mailing address of the ISA/CA Canadian Intellectual Property Office Place du Portage I, C114 - 1st Floor, Box PCT 50 Victoria Street Gatineau, Quebec K1A 0C9 Facsimile No.: 001(819)953-2476	Authorized officer Reid Mulligan 819-934-7566

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	EP 1,154,356 (Vermeulen) 14 November 2001 (14.11.01) entire document	15
A	entire document	12 to 14

Patent Document Cited in Search Report	Publication Date	Patent Family Member(s)	Publication Date
US 6,286,032	04-09-2001	US 6,286,032 B1	04-09-2001
US2001/027450	04-10-2001	EP 1,139,199 A2	04-10-2001
		JP 2001282619 A	12-10-2001
US 2003/120647	NONE		
US 2004/220975	04-11-2004	AU 2004214014 A1	02-09-2004
		CA 2,516,741 A1	02-09-2004
		CN 1777853 A	24-05-2006
		EP 1,595,197 A2	16-11-2005
		JP 2006518508T T	10-08-2006
		WO 2004/74968 A2	02-09-2004
EP 1,154,356	14-11-2001	EP 1,154,356 A1	14-11-2001